CPUT – Combinatorial-Based Prioritization for User-Session-Based Testing of Web Applications

Sreedevi Sampath, University of Maryland, Baltimore County
Renee Bryce, Utah State University
Sachin Jain, University of Maryland, Baltimore County
Schuyler Manchester, Utah State University
Richard Kuhn and Raghu Kacker, NIST

Overview of CPUT

- Automated and general framework to
  - Create and manage regression test suites
  - Designed for web applications
  - Logs usage data and converts to test cases
  - Prioritizes test cases and creates test orders

- Test cases can then be replayed and used to test web systems
Regression Testing

Implement changes (add/delete functionality), remove bugs

Test the new code: Regression Testing

Rerun all existing tests?
Rerun a subset of existing tests?
Rerun tests in a specific order?

Test Prioritization

1. Rerun existing tests from V1 to ensure changes did not break functionality
2. Write new tests as necessary to test new functionality

Need for Reliable Web Applications

- Increasing shift of applications to the web
  - E.g., Google Docs
- Huge losses on web site failure:
  tune of millions of dollars per hour
- Large number of failures during maintenance

1. Web Application Development - Bridging the Gap between QA and Development by Michal Blumenstyk

Verify/ATI 2011  Sreedevi Sampath, UMBC
Traditional Software Testing Process

1. Application Representation
2. Application Specification
3. Application Implementation
4. Test Case Generator
5. Replay Tool
6. Test Cases
7. Expected Results
8. Actual Results
9. Oracle
10. Pass/Fail

Hard to obtain when testing applications
Traditional Software Testing Process

User-session-based Testing

User sessions-based Testing


Request Parameter-values

Verify/ATI 2011  Sreedevi Sampath, UMBC
Test prioritization

- Order existing tests based on some criterion to achieve a performance goal
  - Examples of traditional prioritization criteria: total statement coverage, total method coverage
  - Performance goal: find faults quickly in test execution cycle

User-session-based Test Case Prioritization

- Beta Web Application (v.0.9) Deployment
- Log User Requests
- Prioritized test cases
- Oracle
- Create test cases
- Test Cases
- Replay Tool
- Pass/ Fail
Our test prioritization criteria

- Combinatorial-based
  - 2-way

- Count-based
  - Number of requests
  - Number of parameter-values


Test Case 1:
Catalog.java, *item_name*="shirt", *item_weight*="2"

View_Cart.java, *ship_type*="air", *zip*="21250"

- 2-way interactions:
  (1,3) (1,4) (2,3) (2,4)

- **Intuition:** Interactions of parameters set to values on different windows expose faults

- **Criterion:** Give higher priority to tests with larger number of 2-way interactions
Count-based: Request length

Test Case 1:
Catalog.java, item_name=“shirt”, item_weight=“2”
View_Cart.java, ship_type=“air”, zip=“21250”

- Number of requests in test: 2
- Intuition: tests that contain more requests are more likely to reveal faults because they cover a large part of the underlying code
- Criterion: Give higher priority to tests with larger number of requests

Count-based: Parameter-value length

Test Case 1:
Catalog.java, item_name=“shirt”, item_weight=“2”
View_Cart.java, ship_type=“air”, zip=“21250”

- Number of parameter-values in test: 4
- Intuition: tests that set more parameters to values are more likely to reveal faults
- Criterion: Give higher priority to tests with larger number of parameter-values
Empirical study: Results

- Measure the rate of fault detection

<table>
<thead>
<tr>
<th>Application</th>
<th>Best 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calc</td>
<td>{PV-LtoS, 2-way, Weighted-Freq}</td>
</tr>
<tr>
<td>Paint</td>
<td>{PV-LtoS, 2-way, MFPS}</td>
</tr>
<tr>
<td>SSheet</td>
<td>{UniqWin, 2-way, 1-way}</td>
</tr>
<tr>
<td>Word</td>
<td>{PV-LtoS, 2-way, MFPS}</td>
</tr>
<tr>
<td>Book</td>
<td>{APS, 1-way, 2-way}</td>
</tr>
<tr>
<td>CRM</td>
<td>{2-way, 1-way, PV-LtoS}</td>
</tr>
<tr>
<td>Maslap</td>
<td>{Weighted-Freq, Action-LtoS, 2-way}</td>
</tr>
</tbody>
</table>

Prioritization criteria that were the best in all the subject applications (3 web and 3 GUI applications). Underlining indicates criterion that is always among the best.

CPUT components: 3 engines

- Logger
- Test case creator
- Beta Web Application (v.0.9) Deployment
- Web Application Implementation (v.1.0)
- Oracle
- Pass/ Fail
- Log User Requests
- User Sessions
- Prioritize test cases
- Replay Tool
- Create test cases
- Actual Results
- Expected Results
- Test Cases
- Test Cases Prioritized

Verify/ATI 2011 Sreedevi Sampath, UMBC
CPUT components

- Three main engines
  - Logger
  - Test case creation
  - Prioritizer

Logger

- Enables capture of HTTP GET and POST requests and associated data
- Developed as a module for Apache web server
- Will work on both Linux and Windows platforms
- Minimal performance overhead
- Easy integration with rest of the web server
Deploying the Logger

- Server administrator must place module with other Apache modules
- Enable the module in Apache’s configuration file (can specify path and filename). Log written to a text file
- Format of entry in log file

[Fri Jan 08 13:33:01 2010] # 127.0.0.1 # POST # /schoolMateApplication/index.php #
PHPSESSID=a9099cd16db2e4134200cd69f6dc87cf #
http://localhost:8000/schoolMateApplication/index.php #
PostData:page2=5&logout=1&page=1

CPUT components

- Three main engines
  - Logger
  - Test case creation
  - Prioritizer
Test case creation (parser)

- Efficient storage and retrieval of web usage logs
  - Store usage log and test cases in PostGreSQL database
- Apply previously proposed heuristics to create test cases [Sampath TSE07, Sprenkle ASE05]
- Extensible and generic format of test cases to enable replay
  - Design an XML format for tests

Test case creation (parser)

- Storing usage log in database
  - Create new table
  - Append to existing table
  - Overwrite existing table
Test case creation: XML test case format

```
<testSuite>
  <session id="1.XML">
    <url>
      <request type>POST</request type>
      <baseurl>/SchoolMate/index.php</baseurl>
      <param>
        <name>book_name</name>
        <value>java</value>
      </param>
      <param>
        <name>book_author</name>
        <value>savitch</value>
      </param>
    </url>
  </session>
</testSuite>
```

CPUT components

- Three main engines
  - Logger
  - Test case creation
  - Prioritizer
**Prioritizer**

- Three prioritization criteria
  - 2way, request-length, parameter-value length
  - Random prioritization

- Accommodate two types of web systems
  - Unique URL: each page is identified by a unique base URL
  - Non-unique URL: application has the same base URL for all its pages, and the value of one or more of parameters is used to determine which page to load next

<table>
<thead>
<tr>
<th>Unique URL</th>
<th>Non-unique URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each page in web application has a unique base URL</td>
<td>Same base URL for all the web pages. Value of parameter-values determines unique page</td>
</tr>
<tr>
<td>Example test case</td>
<td>Example test case</td>
</tr>
</tbody>
</table>

- **Example test case**
  - index.php
  - registration.php?name=tom
  - adduser.php?name=henry&pass=joy

- **Example test case**
  - index.php?name=tom&page1=0&page2=1
  - index.php?name=henry&pass=joy&page1=1&page2=1
Tool Demonstration

CPUT main screen